FeRG-LLM : Feature Engineering by Reason Generation Large Language Models

Jeonghyun Ko[1*], **Gyeongyun Park**[1*], **Donghoon Lee**[1*], **Kyunam Lee**[2*†]

Korea University[1], SK Telecom [2]

{2014jhyun, pk53ar, ydh82066}@gmail.com, kyunam@sk.com

Abstract

One of the key tasks in machine learning for tabular data is feature engineering. Although it is vital for improving the performance of models, it demands considerable human expertise and deep domain knowledge, making it labor-intensive endeavor. To address this issue, we propose a novel framework, **FeRG-LLM** (**Fe**ature engineering by **R**eason **G**eneration **L**arge **L**anguage **M**odels), a large language model designed to automatically perform feature engineering at an 8-billion-parameter scale. We have constructed two-stage conversational dialogues that enable language models to analyze machine learning tasks and discovering new features, exhibiting their Chain of Thoughts (CoT) capabilities. We use these dialogues to fine-tune Llama 3.1 8B model and integrate Direct Preference Optimization (DPO) to receive feedback improving quality of new features and the model's performance. Our experiments show that FeRG-LLM performs comparably to or better than Llama 3.1 70B on most datasets, while using fewer resources and achieving reduced inference time. It outperforms other studies in classification tasks and performs well in regression tasks. Moreover, since it does not rely on cloud-hosted LLMs like GPT-4 with extra API costs when generating features, it can be deployed locally, addressing security concerns.

1 Introduction

While deep learning has shown remarkable performance on diverse structure of datasets and gained significant attention, classical machine learning models continue to play a pivotal role across various industries to satisfy interpretability requirements and accommodate computational constraints. These models help organizations make data-driven decisions and build a specialized database of newly discovered key features that significantly enhance model performance. However, uncovering these features requires considerable expertise and analytical skills on the part of data scientists, creating a strong demand for highly qualified professionals.

Since recent large language models (LLM) have achieved superior performance (Achiam et al., 2023; Dubey et al., 2024) in various evaluation metrics due to their extensive knowledge and advanced text-understanding capabilities, several studies have applied language models to feature engineering (Hollmann et al., 2024; Han et al., 2024). They have successfully proposed methodologies for classification tasks, but they rely on cloud-hosted LLMs such as OpenAI GPT (Achiam et al., 2023) so that it cannot be used for sensitive data and remain limited to classification tasks.

Therefore, we propose **FeRG-LLM**, a novel framework that optimizes a language model to discover valuable features for machine learning models. We aim for the language model to fully understand machine learning tasks related to tabular data and generate meaningful features that improve performance. To this end, we construct a dialogue dataset that thoroughly captures the task requirements and yields meaningful features. In the next stage, we train the 8B-scale Llama 3.1 model (Dubey et al., 2024) on these dialogues and then aligned our model using Direct Preference Optimization (DPO) to linguistically generate a more refined rationale for feature creation. Figure 1 presents the overall framework.

Considering that binary classification tasks are critical in industry, we extensively evaluate our model on 14 relevant datasets. Despite being 8B-scale, the results show that our model achieves performance comparable to that of a 70B model and outperforms other recent studies. In addition

[*] equal contribution.

[†] corresponding author.

Findings of the Association for Computational Linguistics:
NAACL 2025, pages 4211–4228

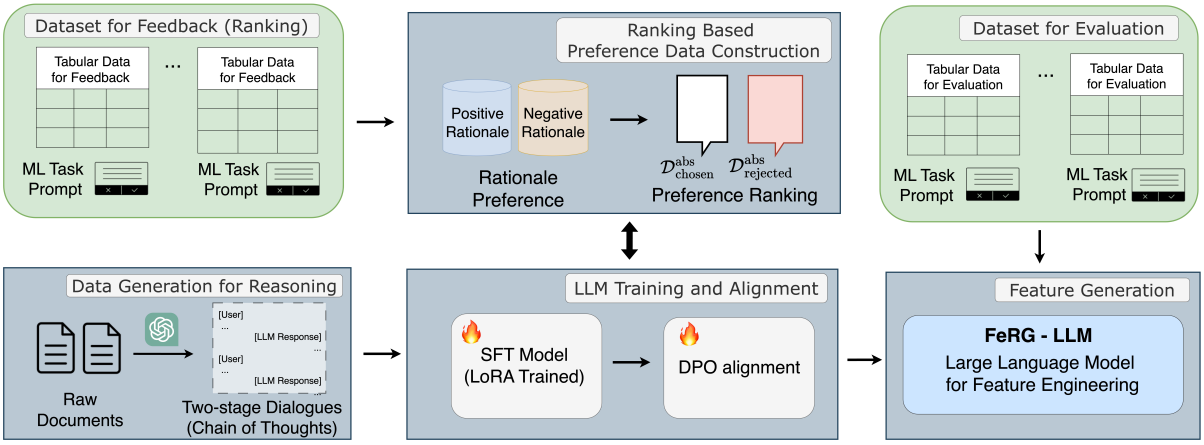

Figure 1: Overall framework of FeRG-LLM. The method first generates a two-stage dialogue related to feature discovery and then performs SFT using LoRA. It then optimizes the language model to provide reasoning feedback.

to experiments on classification tasks, we demonstrate that our model can be readily applied to other problem types, such as regression, using four additional datasets.

Our contributions are threefold:

- We propose a novel framework, **FeRG-LLM**, that harnesses chain of thought reasoning through a two-stage dialogue process in a large language model and integrates feedback via DPO to better understand machine learning tasks on tabular data so that valuable features are discovered.
- Our framework uses Llama 3.1 8B which can be deployed on a single GPU so that it can be used in local environments. Nonetheless, it matches or sometimes even exceeds the capabilities of a 70B model and related work. It greatly reduces resource requirements and eliminates data security concerns making it highly suitable for enterprise adoption.
- Our method produces executable Python code for feature generation, eliminating manual effort and identifying overlooked patterns. By leveraging only variable names and descriptions, it formulates features without distribution analysis. This automation accelerates data-driven decisions while boosting overall efficiency.

2 Related Work

Feature Engineering with Language Models There have been studies of LLMs in the context of feature generation problems for tabular datasets. For instance, CAAFE (Hollmann et al., 2024) provides OpenAI GPT with information about a specific Machine Learning (ML) problem including dataset information and prompt it to suggest new features, along with Python code and rationales. It also incorporate feature selection based on features proposed by GPT. However, this approach focuses on small-scale datasets and requires including variable descriptions, specific values, and details about missing data in the prompt. This approach makes working with large-scale datasets difficult and handling security issues challenging, limiting its applicability in corporate or academic research.

Another study by Han et al. (2024) introduces FeatLLM which creates rules to generate binary features from given data using a few-shot approach. It provides specific data values and corresponding labels to OpenAI GPT to generate binary feature representations for classification. Generated features are used in a linear layer to build a classification model. However, its generation of only binary features may limit the diversity of feature engineering and offer limited linguistic interpretability.

We compare the capabilities of FeRG-LLM with these models as outlined in Table 1 and found that it exhibits several advantages over other models.

Chain of Thought Guiding models to solve complex problems through step-by-step reasoning similar to human thinking is referred to as Chain of Thought (CoT), and it has become a significant area of research in the field of LLMs. The study (Wei et al., 2022) shows that large language models possess exceptional problem-solving abilities

Criteria	CAAFE	FeatLLM	**FeRG-LLM**
Create contextual features by understanding the dataset and modeling objectives?	✓	✓	✓
Operate without explicitly specifying operators such as inequalities in the prompt?	✓	✗	✓
Applicable to tasks beyond classification (e.g., regression)?	✗	✗	✓
Produce features at the point of generation without external API like GPT-4?	✗	✗	✓
Generate features without requiring specific variable values in the prompt?	✗	✗	✓

Table 1: The capabilities of FeRG-LLM are compared with those of other recent models including CAAFE (Hollmann et al., 2024) and FeatLLM (Han et al., 2024). A ✓ indicates that a criterion is met while a ✗ indicates that it is not.

on complex tasks by generating intermediate reasoning steps in text, rather than solely predicting the final answer.

Meanwhile, Magister et al. (2022) notes that CoT is less effective for smaller models. They demonstrate that training smaller models on reasoning patterns from larger models can partially succeed in reproducing CoT. However, they emphasize that this approach relies on the performance of teacher model, making it challenging to enhance the reasoning capability of smaller models.

Nevertheless this study provides insights into the potential for achieving CoT capabilities even in smaller models by focusing on specific tasks. Building on these insights, we employ a two-stage dialogue framework to achieve CoT abilities. This approach fosters a robust understanding of complex machine learning problems and helps uncover meaningful features.

Direct Preference Optimization Rafailov et al. (2024) introduces Direct Preference Optimization (DPO) a method for aligning language models by utilizing human preference data. In contrast the approach of training language models using the Reinforcement Learning from Human Feedback (RLHF) method as proposed in Bai et al. (2022); Ouyang et al. (2022) requires training a reward model before optimizing a language model using the PPO algorithm (Schulman et al., 2017). DPO directly optimizes language models by leveraging preference data, eliminating the need to train a reward model. The preference dataset $\mathcal{D} = \{(x^{(i)}, y_l^{(i)}, y_w^{(i)})\}_{i=1}^N$ is collected from human annotators who rank responses by preference, where y_l represents the less preferred response and y_w the more preferred one.

Studies have extensively investigated the implementation of DPO in LLMs. Allam (2024) proposes BiasDPO, designing DPO training to prioritize the generation of unbiased text on sensitive topics such as gender, race, and religion. They demonstrate that this approach can effectively and reliably mitigate bias. Yang et al. (2024) demonstrates that by assigning different weights to responses based on the strength of preferences, it is possible to generate responses that better reflect human preferences. Zeng et al. (2024) improves the accuracy of language models in writing and language generation by optimizing at the individual token level, as opposed to the traditional methods that optimize at the sentence or task level.

For our study, we apply DPO instead of RLHF, as it enables automatic feedback on generated rationale in resource-constrained settings. This ensures stable feature generation, boosting machine learning performance.

3 Methodology

In this section, we present our FeRG-LLM framework. Feature engineering is not about finding a single correct solution. It requires deep data understanding and the ability to derive meaningful insights through ML and domain expertise. To equip LLMs with such capabilities, we construct a two-stage dialogue dataset for supervised fine-tuning and utilize DPO to offer feedback on the rationale behind feature generation. Furthermore, our framework is designed to efficiently handle large-scale datasets with numerous features and samples, ensuring flexibility and scalability. For space constraints, we provide detailed SFT and DPO training setups, including learning rate, configuration, and other settings, in Appendix D.

3.1 Two-stage Dialogue for Feature Engineering

Two-stage Dialogue Generation Two-turn dialogue dataset is constructed to leverage the language model's CoT reasoning capabilities using

[USER]

```
domain : <domain information>
type: <classification/regression/etc>
present variable in dataframe:
<feature1 name> : <feature1 description>
<feature2 name> : <feature2 description>
⋮
Provide key ideas for creating new derived
variables to improve the performance of the
following ML problem using the given dataset.

Machine Learning PROBLEM: <purpose of data
analysis>
```

[LLM Assistant]

```
1. definition: 1st new feature definition...
            - 1st new feature description...

2. definition: 2nd new feature definition...
            - 2nd new feature description...

                    ⋮
```

[USER]

```
Create new features in Python code to enhance
machine learning performance using the ideas
you provided.
```

[LLM Assistant]

```
# 1. 1st new feature.
  df['1st new'] = ...

# 2. 2nd new feature.
  df['2nd new'] = ...

  ⋮
```

Figure 2: Example of two-stage dialogue that facilitates the LLM's reasoning process. The first step involves conceptualizing core ideas, followed by the second step, where these ideas are actualized through the creation of Python code.

about 7,190 documents, including papers and materials with data analysis or modeling. GPT-4o-mini API is employed to extract the ML-relevant attributes and contextual task information from each document which is then used to construct a conversational dataset.

The extracted information involves identifying the data domain, describing the variables, clarifying the machine learning problem and its type such as regression or classification, and detailing any innovative features. The specific API prompts used are provided in Appendix A.

Reasoning Process We reformat the API-extracted information into a two-step dialogue structure to effectively leverage CoT reasoning abilitiy in 8B-scale LLM. In the first turn, the user provides overall information for feature engineering — domain and variables of dataset, the analysis objective, and the ML problem being addressed. At the end of the prompt, the user explicitly instructs the model to provide key ideas for deriving new features. Then the LLM generates the fundamental ideas for creating new features. It is noteworthy that the prompt presents dataset information solely as variable names and descriptions, thereby relieving data analysts from the need to manually examine distributions of each variable.

In the second turn, the model generates Python code to create new features based on the key ideas proposed in its first response. The specific dialogue prompts are shown in Figure 2. It should be noted that by directly outputting Python code, feature engineering and evaluation can be fully automated. Finally, we carry out supervised fine-tuning methodology LoRA (Hu et al., 2021), obtaining the fine-tuned model π_{SFT}.

3.2 Aligning Language Model with DPO

Preference Feedback To ensure FeRG-LLM generates stable and high-quality features, we perform feature engineering with π_{SFT} on eight tabular datasets designed for binary classification. These datasets are selected to reflect diverse aspects of ML tasks, forming a robust evaluation set. Detailed information is provided in Appendix E. As is common among data scientists, we use Area Under the Curve (AUC) as the reference metric. Based on this, we provide feedback on the generated rationales.

Preference Data Composition For each dataset, we begin with a prompt x as shown in Figure 2 that contains comprehensive information about the machine learning task. Rationale responses R are sampled from $\pi_{\text{SFT}}(\cdot|x)$. The language model subsequently produces final features F, and we compute the AUC Z using XGBoost (Chen and Guestrin, 2016) when the features F are included. During inference, we set the temperature to 1.4 and top-p to 0.9. We set this higher temperature to encourage broader feature exploration while ensuring that the model consistently provides rationales and Python code as intended. This process is repeated for 30 iterations, yielding the sets of rationales $\{R_i\}_{i=1}^{30}$ and features $\{F_i\}_{i=1}^{30}$. In fact, we use all combinations of the generated rationales to create preference data, as all cases are explained in the following description.

For each iteration, we compare the AUC value

Z obtained with the derived feature to the baseline AUC_{base} from fitting XGBoost to the original dataset. If Z is greater than AUC_{base}, the corresponding output is stored in

$$\mathcal{D}^{\text{pos}}_{\text{dataset}} = \{(R^{\text{pos}}_1, F^{\text{pos}}_1, Z^{\text{pos}}_1), \dots\}$$

otherwise, it is stored in

$$\mathcal{D}^{\text{neg}}_{\text{dataset}} = \{(R^{\text{neg}}_1, F^{\text{neg}}_1, Z^{\text{neg}}_1), \dots\}$$

We consider two specific criteria: absolute and relative preferences.

- **Absolute preference** Within each dataset, we form all possible combinations of positive and negative rationale pairs using instances from $\mathcal{D}^{\text{pos}}_{\text{dataset}}$ and $\mathcal{D}^{\text{neg}}_{\text{dataset}}$. In each pair, the R^{pos} included in $\mathcal{D}^{\text{pos}}_{\text{dataset}}$ is labeled as "chosen," while the R^{neg} included in $\mathcal{D}^{\text{neg}}_{\text{dataset}}$ is labeled as "rejected."
- **Relative Preference** We form pairs of rationales exclusively within each of $\mathcal{D}^{\text{pos}}_{\text{dataset}}$ and $\mathcal{D}^{\text{neg}}_{\text{dataset}}$. For each pair we compare their AUC scores Z. The rationale with the higher score is labeled "chosen" and the other "rejected."

Since every possible combination of rationales generated by the LLM falls under one of the two criteria, we construct the preference dataset using all pairs. Finally, we define the preference dataset as $\mathcal{D} = \{(x^{(i)}, r^{(i)+}, r^{(i)-})\}_{i=1}^{M}$, where r^+ represents the chosen rationale, r^- represents the rejected rationale and M is the number of all possible combinations. We optimize the language model π_θ to minimize the objective from Rafailov et al. (2024):

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{SFT}}) = -\mathbb{E}_{(x,r^+,r^-)\sim\mathcal{D}}\left[\log \sigma\Big(p(r^+ \mid x) - p(r^- \mid x)\Big)\right]$$

where $p(\cdot|x) = \beta \log \frac{\pi_\theta(\cdot \mid x)}{\pi_{\text{SFT}}(\cdot \mid x)}$ and $\sigma(\cdot)$ is a logistic function. The parameter β, which we set to 0.1 during training, controls the deviation from π_{SFT}.

4 Experiments

4.1 Expermiental Setup

To evaluate how effectively FeRG-LLM performs feature engineering, we design a method that mirrors the process data scientists follow when discovering new features. Since they refine feature through trial and error, our approach replicates their workflow, enabling us to assess the feature generation capability of LLM. The hardware setup is introduced separately in Section 4.5, along with an analysis of inference time.

AUC evaluation Method We use various tabular datasets, each with its own ML objective, to evaluate the quality of our model-generated features. For each evaluation dataset, the experiment is repeated 7 times, measuring the mean and standard deviation for each experiment as detailed in the pseudocode provided in Algorithm 1. For binary classification tasks, we employ XGBoost for model fitting using AUC as the evaluation metric based on the ROC curve. When fitting the ML model, we split the data into training and testing sets at an 8:2 ratio and perform simple hyperparameter tuning for XGBoost.

Algorithm 1 AUC Evaluation Method

```
Let N ← {# of experiments}
Let L ← {Max iterations without improvement}
Let K ← {Required improvements}
Calculate A_basic using XGBoost on the dataset {Baseline AUC}
for i ← 1 to N do
    Initialize c ← 0, t ← 0
    Initialize set S_i ← ∅ {Set to store improved AUCs}
    while c < K and t < L do
        Obtain A_curr from model
        if A_curr > A_basic then
            Add A_curr to S_i
            c ← c + 1
            t ← 0
        else
            t ← t + 1
        end if
    end while
    Save S_i for experiment i
end for
Collect the maximum AUC values from S_1, S_2, ..., S_N (i.e., A_max,i = max(S_i)) into a set S, then compute its mean and standard deviation.
```

Experiment Execution We utilize FeRG-LLM with a prompt containing dataset and ML task information, setting the temperature to 1.4 and top-p parameter to 0.9. The model generated Python code for feature creation, which was subsequently executed to produce a dataset with the newly added features.

If at most five features are generated we perform a exhaustive search to evaluate AUCs for all combinations and select the one with highest. For six or more, we evaluate each one individually as well as the entire set for efficiency. This procedure is

Method	Dataset						
	Bank	**Nhanes**	**Diabetes**	**Cultivar**	**Credit**	**Ethereum**	**Churn**
XGB	93.19	75.08	79.28	70.64	81.65	87.91	92.12
CAAFE	93.21±0.32	74.47±0.43	81.29±1.28	71.27±2.28	82.67±1.65	87.94±0.17	92.45±0.51
FeatLLM	80.05±0.62	66.57±0.71	78.66±0.47	69.52±2.23	50.64±7.19	91.22±0.73	79.91±0.79
Llama 3.1 70B	93.36±0.05	**76.30±0.42**	**83.12±0.38**	76.38±0.38	84.82±0.27	**97.43±0.29**	**93.25±0.27**
FeRG-LLM	**93.88±0.51**	76.23±0.40	82.43±0.34	**78.34±2.74**	**85.07±0.67**	97.03±0.60	93.22±0.14

Method	Dataset						
	Ad-click	**Aging-npha**	**Infrared-t**	**Ilpd-indian**	**Support2**	**Heart-failure**	**Glioma**
XGB	63.85	56.97	85.74	79.48	85.28	88.57	91.13
CAAFE	65.41±0.65	58.42±1.94	86.30±0.44	79.48±0.00	85.42±0.05	87.67±0.62	91.11±0.25
FeatLLM	51.18±0.57	59.16±0.75	80.48±0.53	82.61±1.17	56.90±7.96	80.77±1.08	73.51±3.28
Llama 3.1 70B	70.67±0.61	**61.47±0.83**	**87.49±0.28**	82.94±0.94	85.72±0.06	89.66±0.29	**92.04±0.44**
FeRG-LLM	**81.55±8.64**	60.38±0.78	87.42±0.27	**83.83±0.51**	**85.76±0.16**	**90.17±0.33**	91.75±0.30

Table 2: The overall mean AUC and standard deviation for major models, including FeRG-LLM, are presented. XGB, the baseline model (Chen and Guestrin, 2016), achieved an AUC of 80.78 without feature engineering. After each model generated features for 14 binary classification tasks, AUC scores were compared : CAAFE scored 81.22, FeatLLM 71.51, Llama 3.1 70B 83.9, and FeRG-LLM achieved the highest at 84.79. For FeatLLM, MICE (Van Buuren and Groothuis-Oudshoorn, 2011) was applied to handle missing values, as recommended in the original paper.

repeated until K [1] improved AUC outputs are observed and the best value is recorded. The model terminates after L iterations if there is no improvement.

Datasets We prepare datasets from the UCI Repository, representing standard machine learning problems, including Bank Moro and Cortez (2014), Nhanes UCI Machine Learning Repository (2019), Cultivar Rodrigues de Oliveira and Mario Zuffo (2023), Ilpd-indian Ramana and Venkateswarlu (2022), Glioma (Tasci and Zhuge, 2022), Support2 Knaus et al. (1995), and Heart-failure UCI Machine Learning Repository (2020). Additionally, we obtain datasets from Kaggle, including Diabetes, Credit, Ethereum, Churn, and Ad-click, to assess their applicability in real-world classification tasks. Also, there is a potential risk that the dialouge dataset might contain direct information about the evaluation sets. To address this, we additionally include the recent datasets : Aging-npha (UCI Machine Learning Repository, 2017) and Infrared-t (Wang et al., 2021) from the UCI Repository. Detailed information on each dataset, including data size, label ratio, and the specific machine learning problem, is provided in Appendix F.

[1]We conduct experiments setting $N = 7$, $L = 15$, and $K = 3$.

4.2 Main Results

Baselines We evaluate our method with CAAFE (Hollmann et al., 2024), FeatLLM (Han et al., 2024) and Llama 3.1 70B. In the original framework of CAAFE, TabPFN (Hollmann et al., 2025) is used for model training and providing criteria such as AUC and ACC for feature selection. However, we replace it with XGBoost because TabPFN struggled to handle large datasets such as Bank and Ethereum, and often fell short compared to XGBoost in various experiments. FeatLLM generates features with a GPT-based API using 16 data points and trains a linear layer on up to 128 data points. To further investigate the model's capabilities, we test a setup where FeatLLM generated features using 64 data points instead of 16, and the linear layer is trained on the entire dataset instead of 128 data points. After evaluating multiple configurations, we report the highest observed AUC. Furthermore, we include the Llama 3.1 70B model, which has approximately nine times more parameters than the 8B model, offering significantly greater capacity for reasoning and knowledge retention. This inclusion allows for a meaningful comparison between a large-scale model and a more resource-efficient fine-tuned alternative. All results are presented in Appendix C.

Performance Comparison Table 2 presents overall AUC evaluation results. FeRG-LLM

ML task Prompt (NHANES)	[Domain] Medical [Type] classification [Present variable in dataframe] riagendr: Respondents Gender, paq605:If the respondent engages in moderate or vigorous-intensity sports, fitness, or recreational activities in the typical week, bmxbmi: Respondents Body Mass Index, lbxglu: Respondents Blood Glucose after fasting, diq010:If the Respondent is diabetic, lbxglt: Respondents Oral , lbxin: Respondents Blood Insulin Levels Provide key ideas for creating new derived variables to improve the performance of the following ML problem using the given dataset. [Machine Learning Problem] for predicting the given person's age group from the record.
Without DPO alignment Rationale	definition: interaction term between engagement in activities and bmi, useful to assess the joint effect of these two features on aging risk. consider relationships between physical activity (as indicated by paq605), body metrics (such as bmxbmi), and glucose levels (lbxglu) could help model aging risks as they interact to influence age-related factors. definition: average glucose levels adjusted for diabetic status, helping to evaluate the diabetic effects on glucose. interaction features can provide insights into how these different metrics relate to aging, enabling improved predictions.
Python code	df['activity_bmi_interaction'] = df['paq605'] * df['bmxbmi'] df['adjusted_avg_glucose'] = df['lbxglu'].where(df['diq010'] == 1, 0)
AUC	**75.31**
FeRG-LLM Rationale	definition: the ratio of body mass index (bmi) to blood glucose levels, which might indicate weight management relative to glucose status. the features represent a combination of health metrics that can be interrelated. for example, body mass index (bmi) in relation to blood glucose and activity levels might shed light on overall health status more effectively than examining each variable in isolation. definition: the average of bmi and blood glucose levels to understand overall health status more effectively. interaction between bmi and blood glucose might provide insights into the relationship between weight and glucose metabolism. definition: the ratio of activity levels to blood glucose, depicting how physical activity might relate to glucose status. ratios or combinations of certain health metrics may better capture the nuances of an individual's health profile, which could improve prediction accuracy for age groups.
Python Code	df['bmi_glucose_ratio'] = df['bmxbmi']/df['lbxglu'] df['average_health_metric'] = (df['bmxbmi'] + df['lbxglu'] + df['diq010'].replace('yes':1, 'no':0))/3 df['activity_glucose_ratio'] = df['paq605']/df['lbxglu']
AUC	**76.75**

Table 3: Linguistic Analysis of Reasoning Shift Induced by Rationale Alignment. In the NHANES task, examining the features and their corresponding evidence generated using variables such as paq605, bmxbmi, lbxglu, and diq010 reveals significant changes before and after DPO application.

achieves the highest performance on 7 datasets and the second-highest on the remaining 7, surpassing other recent models. Notably, it performs on par with the 70B model and even outperforms it in Ad-click and Cultivar task. This implies that FeRG framework can deliver performance comparable to that of larger language models while operating with fewer resource constraints and at greater speed, as shown in figure 4. Meanwhile, relatively low AUC have at times been identified in FeatLLM, suggesting that a single linear layer may be insufficient to capture the underlying structure of the data. Conclusively, unlike prior research models that rely on external APIs to generate features, our model functions locally, ensuring data security in corporate environments.

4.3 Reasoning Shift Analysis

We study the effect of providing feedback on generated reasoning by visualizing representations of rationales through t-SNE, leading to linguistic analysis of reasonsing shfit.

Linguistic Analysis of Response We conduct a linguistic analysis of the feature generation rationale and the final code as shown in Table 3. For NHANES task, FeRG-LLM and the model prior to DPO both adopt the same four variables paq605, bmxbmi, lbxglu, and diq010, yet yield entirely distinct outputs. This shift improve AUC from 75.31 to 76.75. We also confirm that the Python code is derived directly from the rationale, indicating a strong link between the explanation and final implementation. More examples of FeRG-LLM's responses are in Appendix B.

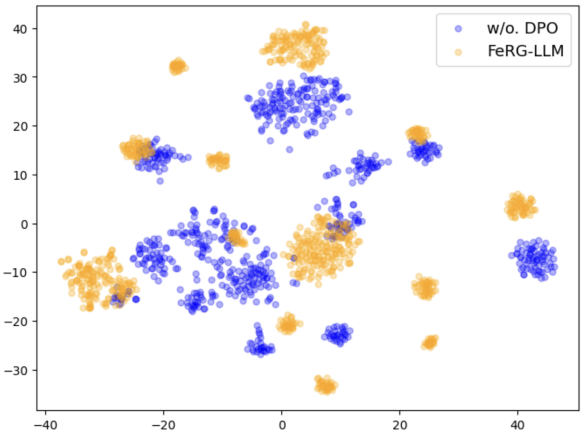

Figure 3: t-SNE Visualization of Reasoning Embeddings. Yellow points represent embeddings of reasoning generated by FeRG-LLM, while blue points correspond to those generated without DPO.

Embedding Shift To explore the vector representations of each response, we use OpenAI text-embedding-3-small model. As shown in Figure 3, the reason distribution without alignment (blue) and that of FeRG-LLM (orange) exhibit distinct shifts in reasoning patterns. After applying DPO feedback, new content emerged where none had existed before, while previously produced content converged into a more narrowly confined space.

Ablation Study Table 4 presents the ablation results for FeRG-LLM across 14 datasets. In every case, generating features based on rationales contributes to superior performance compared to their absence. On average, the model that immediately produces Python code exhibits a 1.06% lower AUC than its rationale-driven counterpart. In addition, the effectiveness of alignment is quantitatively confirmed with a 0.34% AUC gain. While this improvement may seem modest, the consistently better performance across most datasets suggests that DPO meaningfully enhances the expressiveness of the generated features.

Dataset	w/o. Rationale	w/o. DPO	FeRG-LLM
Bank	93.69±0.44	93.40±0.10	**93.88±0.51**
Nhanes	75.84±0.39	75.73±0.34	**76.23±0.40**
Diabetes	82.25±0.51	82.25±0.29	**82.43±0.34**
Cultivar	76.57±2.36	**79.94±5.24**	78.34±2.74
Credit	83.93±0.59	84.11±0.38	**85.07±0.67**
Ethereum	96.76±0.50	96.78±0.92	**97.03±0.60**
Churn	93.04±0.45	93.03±0.30	**93.22±0.14**
Ad-click	76.22±9.72	80.50±12.65	**81.55±8.64**
Aging-npha	59.85±1.44	60.36±1.13	**60.38±0.78**
Infrared-t	87.12±0.28	87.20±0.23	**87.42±0.27**
Ilpd-inidan	82.26±1.02	82.73±0.76	**83.83±0.51**
support2	85.66±0.10	85.65±0.11	**85.76±0.16**
heart-failure	89.63±0.40	89.79±0.19	**90.17±0.33**
giloma	91.58±0.12	91.53±0.07	**91.75±0.30**
Mean	83.89 (-1.06%)	84.50 (-0.34%)	**84.79 (–)**

Table 4: Ablations of each component of FeRG-LLM : Rationle prompting and dpo alignment process.

4.4 Beyond Classification Task

Dataset	XGBReg	w/o. DPO	**FeRG-LLM**
Bike	1408.80	1368.63 ± 9.50 (-2.85%)	1337.76 ± 35.35 (-5.04%)
Concrete	18.61	17.18 ± 0.66 (-7.68%)	16.47 ± 0.58 (-11.50%)
Parkinsons	3.04	2.69 ± 0.09 (-11.51%)	2.76 ± 0.05 (-9.21%)
Realestate	32.60	30.18 ± 0.94 (-7.42%)	29.26 ± 0.81 (-10.24%)

Table 5: Overall Mean MSE and Standard Deviation for Regression Tasks Using FeRG-LLM.

We further highlight that our framework can be extended to other tasks. Specifically we apply FeRG-LLM to regression datasets, details in Appendix G, by switching the evaluation metric from AUC to MSE and replacing XGBClassifier with XGBRegressor. Table 5 presents a 5–10% decrease in MSE across all datasets compared to cases without feature engineering. Also, FeRG-LLM generally outperforms a non-rationale baseline on regression tasks—except for the Concrete—demonstrating the efficacy of DPO-driven

feedback. Hence, our approach is not limited to classification tasks but has the potential to improve performance over a wider spectrum of predictive problems.

4.5 Inference Speed

In Section 4.2, FeRG-LLM and Llama 3.1 70B alternate in achieving top performance. Therefore, to further assess FeRG-LLM's efficiency, we measure inference speed by generating 20 outputs for each of the 12 datasets, resulting in a total of 240 runs. This approach ensures that the results are relatively consistent across various datasets. The Llama 3.1 70B model run on an AMD EPYC 7513 system with 503 GB of RAM in two configurations: four RTX A6000 GPUs (48GB VRAM each) and two A100 GPUs (80GB VRAM each). FeRG-LLM is tested on a university-shared Intel Xeon Gold 6348 server with 16GB allocated per user and a single RTX A6000. It is also tested on RunPod with the same system specifications as the 70B model, using one RTX A6000 with 48 GB of VRAM.

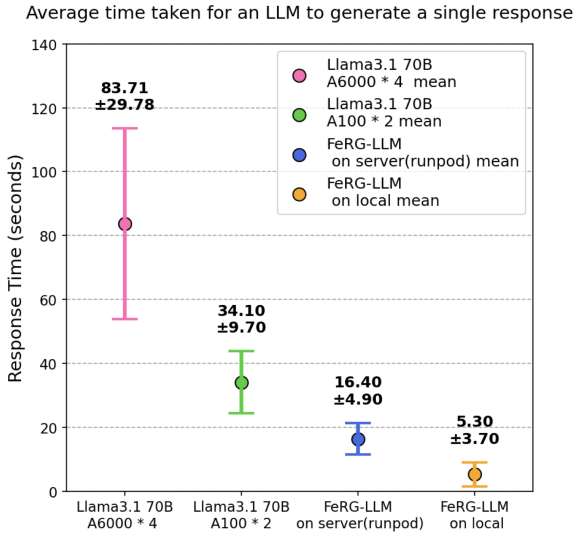

Figure 4: Inference time comparison of the 70B model and FeRG-LLM using different configurations. The 70B model takes 83.71 seconds on four RTX A6000 and 34.10 seconds on two A100, while FeRG-LLM completes inference in 16.4 seconds on a single RTX A6000 and local feature generation in about 5 seconds on a single RTX A6000.

On the cloud, the 70B model takes an average of 83.71 seconds in the A6000 setup and 34.10 seconds in the A100 sutup. But FeRG-LLM is much faster, averaging 16.4 seconds and only 5.3 seconds in non-cloud settings. In summary, given the similar performance of both models and the results from experiments conducted according to Algorithm 1, where FeRG-LLM completed an average of 7.4 runs per experiment compared to 5.7 for the 70B model, our model achieves significantly higher efficiency with fewer GPUs and faster inference time.

5 Conclusion

In this research, we develop a novel framework, FeRG-LLM, an automated feature engineering language model that assists data scientists in discovering features for building effective machine learning models. By optimizing a small LLM (e.g., Llama 3.1 8B), FeRG-LLM can operate efficiently even on a company's limited infrastructure and ensure data security. This study explores LLM capabilities by pushing their limits to assess their potential in automating data-driven tasks.

The proposed framework offers significant practical utility in real-world industrial applications. It is ideal for startups and teams with limited GPU resources and workforce. Even large enterprises can achieve automation and efficiency by applying our model to large-scale datasets with a large number of variables. Also, our model can be easily adapted to diverse tasks by simply replacing the ML prompts with appropriate tasks and evaluation metric. This flexibility makes the framework highly extensible and opens avenues for future research in diverse machine learning contexts.

Limitation

A limitation of our study is the insufficient analysis of the trade-off between computational overhead and performance gains in our two-step chain-of-thought and DPO training process. Although DPO-driven feedback improves performance compared to the model before DPO training, the additional training and inference costs are not fully quantified. Future work should assess these trade-offs to better establish the practical viability of our approach.

From a structural perspective, our model performs two sequential calls to leverage the LLM's Chain-of-Thought capability. However, it currently generates features directly from the initial reasoning output without further refinement. Additionally, our investigation into the optimal step size remains insufficient. These limitations warrant further exploration and improvement in future research.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Ahmed Allam. 2024. BiasDPO: Mitigating bias in language models through direct preference optimization. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pages 71–79, Bangkok, Thailand. Association for Computational Linguistics.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA. ACM.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Sungwon Han, Jinsung Yoon, Sercan O Arik, and Tomas Pfister. 2024. Large language models can automatically engineer features for few-shot tabular learning. In *Forty-first International Conference on Machine Learning*.

Noah Hollmann, Samuel Müller, and Frank Hutter. 2024. Large language models for automated data science: Introducing caafe for context-aware automated feature engineering. *Advances in Neural Information Processing Systems*, 36.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. 2025. Accurate predictions on small data with a tabular foundation model. *Nature*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

William A. Knaus, Frank E. Harrell, Joanne Lynn, Lee Goldman, Russell S. Phillips, Alfred F. Connors, Neal V. Dawson, William J. Fulkerson, Robert M. Califf, and Norman Desbiens. 1995. The support prognostic model: objective estimates of survival for seriously ill hospitalized adults. *Annals of Internal Medicine*, 122(3):191–203.

Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. 2022. Teaching small language models to reason. *arXiv preprint arXiv:2212.08410*.

Rita P. Moro, S. and P. Cortez. 2014. Bank Marketing. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5K306.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Bendi Ramana and N. Venkateswarlu. 2022. ILPD (Indian Liver Patient Dataset). UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5D02C.

Bruno Rodrigues de Oliveira and Alan Mario Zuffo. 2023. Forty Soybean Cultivars from Subsequent Harvests. UCI Machine Learning Repository. DOI: https://doi.org/10.46420/TAES.e230005.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Camphausen Kevin Krauze Andra Valentina Tasci, Erdal and Ying Zhuge. 2022. Glioma Grading Clinical and Mutation Features. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5R62J.

UCI Machine Learning Repository. 2017. National Poll on Healthy Aging (NPHA). UCI Machine Learning Repository. DOI: https://doi.org/10.3886/ICPSR37305.v1.

UCI Machine Learning Repository. 2019. National Health and Nutrition Health Survey 2013-2014 (NHANES) Age Prediction Subset. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5BS66.

UCI Machine Learning Repository. 2020. Heart Failure Clinical Records. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5Z89R.

Stef Van Buuren and Karin Groothuis-Oudshoorn. 2011. mice: Multivariate imputation by chained equations in r. *Journal of statistical software*, 45:1–67.

Quanzeng Wang, Yangling Zhou, Pejman Ghassemi, David McBride, J. Casamento, and T. Pfefer. 2021. Infrared thermography temperature [dataset]. UCI Machine Learning Repository.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Guangyu Yang, Jinghong Chen, Weizhe Lin, and Bill Byrne. 2024. Direct preference optimization for neural machine translation with minimum bayes risk decoding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 391–398.

Yongcheng Zeng, Guoqing Liu, Weiyu Ma, Ning Yang, Haifeng Zhang, and Jun Wang. 2024. Token-level direct preference optimization. *arXiv preprint arXiv:2404.11999*.

A Details : Data Generation from papers

We utilize OpenAI GPT-4o-mini to gather essential information from various papers related to tabular data analysis. Figure 5 illustrates the specific prompt used for data extraction within these studies.
A total of 7 sections are extracted titled 'Domain', 'Type', 'Available Features in dataframe', 'ML Problem', 'Thought for deriving new feature', 'New Features', and 'Python code.', and the final Python code is provided without any additional explanations or text, allowing it to be executed directly. Subsequently, we make numerous revisions to enhance the quality of the data, ensuring its accuracy and reliability.

System Instruction:

You are a highly skilled machine learning engineer. You need to extract the content used to create new derived features to improve actual machine learning performance based on the information in the paper.

1. Read the provided text and organize it into the following sections:'Domain', 'Type', 'Available Features in dataframe', 'ML Problem', 'Thought for deriving new feature', 'New Features', and 'Python code.'

2. Since the focus is on discovering new features applicable to existing tabular data, if there is no appropriate information for each section, refer to previous prompts or simply answer "None" for that section.

3. Include the variable names and their descriptions from the tabular dataset in the 'Available Features in dataframe' section.

4. If new derived features are mathematically defined, ensure the variables in the formula come from the 'Available Features in dataframe' section only.

5. Even if there is no information on new derived features in the text, if you can propose features based solely on the 'ML Problem' and 'Available Features in dataframe,' you may do so in the 'New Features' section, even if it's not from the paper.

6. Include the ideas, knowledge, and expertise used to create new derived features to improve machine learning performance in the 'Thought for deriving new feature' section.

7. Assume the contents in 'Available Features in dataframe' are columns present in a DataFrame 'df,' and convert the formulas in 'New Features' into Python code. Respond with only the Python code. It should be executable.

8. If there are multiple contexts or solutions, record them separately.

Please respond in English.

Figure 5: Data Generation Prompt Using the GPT API.

B More Example of Feature Engineering with FeRG-LLM

The results of FeRG-LLM's feature engineering for Ethereum and the Cultivar task are presented in the table below. As discussed in the main text the model identifies a rationale to guide the discovery of new features and subsequently generates the final Python code based on this rationale.

System Instruction	You are an automated ML engineer who writes Python code based on a given domain, present variables in a dataframe, and ML problems. Whenever the sentence write down python code appears, immediately write Python code.
ML task Prompt (Ethereum)	[Domain] Blockchain [Type] classification [Present variable in dataframe] txhash: Unique Transaction Hash blockheight: The block number in which the transaction was included timestamp: The exact date and time when the transaction was processed from: The sender's Ethereum address to: The receiver's Ethereum address value: The amount of Ether transferred in the transaction Provide key ideas for creating new derived variables to improve the performance of the following ML problem using the given dataset. [Machine Learning Problem] identify patterns and anomalies indicative of fraudulent activity.
FeRG-LLM Rationale	1. definition: total amount transferred by a sender within a given time frame. - aggregated values to assess sender-receiver behavior, which can help identify unusual patterns associated with fraud. 2. definition: count of distinct transactions initiated by a particular sender in a specific time frame. - use of time-based features to uncover patterns over time and evaluate frequency and volume of transactions, allowing for better fraud detection. 3. definition: total number of times a unique sender-receiver pair exchanged values across transactions - utilize transaction volume and frequency data to identify abnormal flows indicative of potential scams.
Python code	# 1. total transaction value sent by each sender over a certain period. df['totalvalue'] = df.groupby('from')['value'].transform('sum') # 2. count of transactions per sender within a specific time frame. df['count_transactions'] = df.groupby('from')['from'].transform('count') # 3. frequency of transaction for sender-receiver pairs. df['frequency_tx'] = df.groupby(['from','to'])['from'].transform('count')
ML task Prompt (Cultivar)	[Domain] Agriculture [Type] classification [Present variable in dataframe] season: 1 or 2 cultivar: Cultivar names repetition: 1, 2, 3 or 4 ph: plant height (cm), determined from the soil surface to the insertion of the last leaf using a millimeter ruler ifp: insertion of the first pod (cm), determined from the soil surface to the insertion of the first vegetable nlp: Number of stems (unit), through manual counting ngp: Number of legumes per plant (unit), through manual counting ngl: Number of grains per plant (unit), through manual counting ns: Number of grains per pod (unit), through manual counting mhg: Thousand seed weight (g), according to the methodology described in Brasil (2009) Provide key ideas for creating new derived variables to improve the performance of the following ML problem using the given dataset. [Machine Learning Problem] for predicting how high the grain yield of this soybean cultivar will be.
FeRG-LLM Rationale	1. definition: average number of pods per plant considering the possible repetition of experiments for different seasons and cultivar replications. • purpose of average biomass measures: the various measurements related to plant size and development can be used to compute derived metrics which might relate more closely to grain yield. 2. definition: variation in the number of grains produced per plant across different experimental replications which can highlight stability and performance. • purpose of pod-to-grain ratios: examining the relationship between the number of pods and grains could provide insights into the efficiency of pod utilization, potentially revealing cultivar performance characteristics.
Python code	df['avgpp'] = (df['ngp'].where(df['repetition'] == 1).fillna(0) + df['ngp'].where(df['repetition'] == 2).fillna(0) + df['ngp'].where(df['repetition'] == 3).fillna(0) + df['ngp'].where(df['repetition'] == 4).fillna(0)) / 4 df['grv'] = df.groupby('cultivar')['ngl'].transform(lambda x: ((x - x.mean())**2).mean())

Table 6: Example responses when Ethereum and Cultivar task are given into FeRG-LLM. Definitions are generated for each inference, followed by the corresponding Python code for new features. Some responses that generate code include comments indicated by "#" while others do not. In both cases the Python code can be executed immediately, ensuring that automation works properly.

C All Experimental Results for CAAFE and FeatLLM

In CAAFE, variable selection is performed using TabPFN instead of XGBoost, and the final dataset is fitted with XGBoost. As discussed in the main text, the model fails to fit properly on the Bank and Ethereum datasets, which contain 45,000 and 254,973 samples, respectively. As a result, AUC and ACC values could not be computed.

Dataset	CAAFE+TabPFN
Bank	NaN
Nhanes	75.01 ± 0.68
Diabetes	80.47 ± 1.11
Cultivar	71.43 ± 4.93
Credit	82.01 ± 0.91
Ethereum	NaN
Churn	92.38 ± 0.39
Ad-click	63.85 ± 0.00
Aging-npha	56.40 ± 0.81
Infrared-t	86.04 ± 0.35
Ilpd-indian	80.53 ± 0.98
Support2	85.38 ± 0.05
Heart-failure	87.63 ± 2.23
Glioma	90.79 ± 0.30

Table 7: Performance results of the CAAFE+tabpfn model on various datasets.

When testing FeatLLM, we use a setup where FeatLLM generated features using 16 data points(shots) or 64, and the linear layer is trained on 128 data points or the entire dataset.

Dataset	All data (16 shots)	All data (64 shots)	128 data points (16 shots)	128 data points (64 shots)
Bank	69.25±1.17	66.62±1.23	80.05±0.62	79.17±0.75
Nhanes	59.47±3.89	60.95±1.23	65.73±0.49	66.57±0.71
Diabetes	78.66±0.47	78.16±0.24	78.45±0.36	78.53±0.38
Cultivar	63.19±2.63	62.21±2.93	69.52±2.23	63.12±3.00
Credit	50.64±7.19	36.30±1.45	36.09±1.12	40.21±2.55
Ethereum	73.19±0.98	75.48±2.05	91.22±0.73	87.61±2.65
Aging_npha	58.52±1.22	59.16±0.75	57.47±1.34	56.03±0.76
Infrared_t	80.48±0.53	79.11±0.81	79.04±0.81	77.03±0.98
Churn	73.71±3.28	77.48±2.38	79.91±0.79	79.38±1.16
Ad_click	51.18±0.57	48.98±0.46	47.72±0.39	47.30±0.59
Ilpd_indian	82.61±1.17	81.74±0.81	80.67±0.71	81.13±0.69
Support2	56.70±7.96	55.74±7.99	47.31±7.68	47.42±4.53
Heart_failure	78.26±0.91	80.77±1.08	77.55±1.28	78.36±1.96
Glioma	70.13±5.57	70.59±5.98	73.51±3.28	65.59±6.87

Table 8: Performance results of the FeatLLM on the various setups.

D Training Setup and XGBoost Parameter Search

D.1 Training Setup

We detail our training configurations for both the supervised fine-tuning (SFT) and Direct Preference Optimization (DPO) settings. For SFT, we use a batch size of 4 and train the model for 5 epochs with a learning rate of 1×10^{-4}. Mixed precision training is performed using BF16, and LoRA is applied with a rank of 8, an alpha of 16, and no dropout (0.0). For both SFT and DPO, we utilize a cosine learning rate scheduler. In the case of DPO, the model is trained with a learning rate of 8×10^{-6} for 4 epochs, and the β parameter is set to 0.1.

Parameter	**SFT**	**DPO**
Batch Size	4	4
Epochs	5	4
Learning Rate	1×10^{-4}	8×10^{-6}
Mixed Precision	BF16	BF16
LoRA Rank	8	8
LoRA Alpha	16	16
LoRA Dropout	0.0	0.0
Scheduler	Cosine	Cosine
β	–	0.1

Table 9: Training configurations for SFT and DPO with a cosine learning rate scheduler.

D.2 XGBoost Parameter Search

For fitting the XGBoost model, we conducted a grid search over the following parameter ranges:

Parameter	**Values**
max_depth	3, 5, 7
learning_rate	0.01, 0.1
n_estimators	50, 100, 200
subsample	0.8, 1.0
colsample_bytree	0.8, 1.0

Table 10: Hyperparameter search space for XGBoost.

E Datasets for Feedback

These datasets contain information on target datasets for which features were generated using the supervised fine-tuned model. Based on the evaluation results, we utilized the final reject and chosen datasets to create the training data used for DPO.

Dataset	Size	# Features	Label Ratio(%)	ML Problem
Default of credit card clients	30000	23	77.9 \ 22.1	Predicting customer default payments in Taiwan by comparing the predictive accuracy of the probability of default across six data mining methods.
In vehicle coupon recommendation	12684	23	56.8 \ 43.2	Predicting whether an individual will accept a recommended coupon under different driving scenarios.
Online news popularity	39644	58	85.2 \ 14.8	Predicting the social media popularity of Mashable articles using a heterogeneous set of features collected over a two-year period.
Online shoppers purchasing intention dataset	12330	17	84.5 \ 15.5	Predicting whether a session led to a purchase.
Polish companies bankruptcy	43405	65	95.2 \ 4.8	Predicting bankruptcy of Polish companies based on financial data, where bankrupt companies were analyzed from 2000 to 2012 and operating companies were evaluated from 2007 to 2013.
Blood transfusion service center	748	4	76.2 \ 23.8	Predicting whether a blood donor will donate blood in March 2007 based on the donor's historical data.
Iranian churn	3150	13	84.3 \ 15.7	Predicting customer churn in the telecom industry based on user activity and subscription patterns over a 12-month period.
Productivity prediction of garment employees	1197	14	50.2 \ 49.8	Predicting the productivity of garment employees based on various factors.

Table 11: This table provides descriptions of the datasets used for DPO Training. The reasons for preference based on the AUC are ordered for each dataset.

F Evaluation Datasets for Classification

Dataset	Size	# Features	Label Ratio(%)	ML Problem
Bank	45,211	16	88.3 \ 11.7	Whether a customer will subscribe to a term deposit.
Nhanes	2,278	7	84.0 \ 16.0	Predicting the given person's age group from the record.
Diabetes	768	8	65.1 \ 34.9	Predicting the presence of diabetes in an individual. (https://kaggle.com/datasets/uciml/pima-indians-diabetes-database)
Cultivar	320	10	50.0 \ 50.0	Predicting how high the grain yield of this soybean cultivar will be.
Credit	1,548	18	88.7 \ 11.3	Predicting whether the applicant is approved or rejected. (https://www.kaggle.com/datasets/rohitudageri/credit-card-details?select=Credit_card_label.csv)
Ethereum	254,973	6	93.9 \ 6.1	Identifing patterns and anomalies indicative of fraudulent activity. (https://www.kaggle.com/datasets/chaitya0623/ethereum-transactions-for-fraud-detection)
Churn	4,250	20	85.9 \ 14.1	Predicting whether a customer is likely to churn based on the other features. (https://www.kaggle.com/datasets/mustafakeser4/bigquery-churn-dataset)
Ad-click	10,000	8	65.0 \ 35.0	Predicting whether a user will click on an online ad based on their demographics, browsing behavior, the context of the ad's display, and the time of day. (https://www.kaggle.com/datasets/marius2303/ad-click-prediction-dataset)
Aging-npha	714	14	70.4 \ 29.6	Predicting the categorized number of doctors a patient has visited based on their health status, sleep issues, and demographic information.
Infrared-t	1,020	33	59.5 \ 40.5	Predicting the oral temperature using the environment information as well as the thermal image readings.
Ilpd-indian liver patient dataset	583	10	71.4 \ 28.6	The prediction task is to determine whether a patient suffers from liver disease based on the information about several biochemical markers, including albumin and other enzymes required for metabolism.
Support2	9,105	42	68.1 \ 31.9	The goal is to determine these patients' 2- and 6-month survival rates based on several physiologic, demographics, and disease severity information.
Heart-failure	299	12	67.9 \ 32.1	Predicting whether a patient will die during the follow-up period.
Glioma	839	23	58.0 \ 42.0	Predicting whether the glioma grade is classified as l̈gg̈(lower grade glioma) or g̈bm̈(glioblastoma multiforme), where 0 represents lgg and 1 represents gbm.

Table 12: This table provides descriptions of the datasets used for experiments of feature engineering. These datasets are intended for solving binary classification tasks.

G Evaluation Datasets for Regression

Dataset	Size	# Features	ML Problem
Bike	17379	13	Predicting count of total rental bikes including both casual and registered.
Concrete	1030	8	Predicting concrete compressive strength.
Parkinsons	5875	19	Predicting clinician's total UPDRS score, linearly interpolated.
Realstate	414	6	Predicting house price of unit area.

Table 13: This table provides descriptions of the datasets used for experiments of feature engineering. These datasets are intended for solving regression problems.